Bewertung von Objektmanagementsystemen für Software-Entwicklungsumgebungen*

Sanjay Dewal, Harry Hormann, Lothar Schöpe
Fachbereich Informatik, Universität Dortmund
Postfach 500500, 4600 Dortmund 50

Udo Kelter, Dirk Platz, Michael Roschewski
Fachbereich Mathematik und Informatik, FernUniversität Hagen
Postfach 940, 5800 Hagen

1 Einführung und Übersicht

Unter den Entwicklern von Software-Entwicklungsumgebungen (**SEU**) herrscht inzwischen weitgehende Einigkeit darüber, daß konventionelle Datenbankmanagementsysteme (**DBMS**) nicht zur Verwaltung der komplexen Daten in einer SEU geeignet sind, sondern daß hierzu sog. Nicht-Standard-DBMS, die wir hier als Objektmanagementsysteme (**OMS**) bezeichnen, erforderlich sind.

In den letzten Jahren sind viele, z.T. sehr unterschiedliche OMSe (für SEUen) entwickelt worden [ScH90]. Für einen SEU-Entwickler ist es daher außerordentlich schwierig, diese OMSe zu bewerten und zu vergleichen und das für eine SEU am besten geeignete OMS auszuwählen.

*Diese Untersuchung wurde im Rahmen des EUREKA-Projekts ESF gefördert.

In diesem Papier wird von der ersten Phase eines Projekts berichtet, in dem mehrere OMSe (DAMOKLES Vers. 2.3, GRAS Vers. 4.12, Object-Base Vers. 1.0, PCTE/OMS Vers. 10.4, ProMod/OMS Vers. 1.0) bewertet wurden. Es werden die auftretenden methodischen Probleme, die Ergebnisse der Bewertungen und die Konsequenzen für das weitere Vorgehen beschrieben. Die wichtigsten Erkenntnisse waren:

- Die etablierten Methoden zur Bewertung konventioneller DBS (i.w. Benchmarks) können wegen der uneinheitlichen Funktionalität und Architektur der OMSe nicht ohne Änderungen zur Bewertung der OMSe verwandt werden.
- Die Definition von OMS-Benchmarks unterscheidet sich sowohl im Abstraktionsniveau als auch im Grad der Komplexität von konventionellen Benchmarks.

  Konventionelle Benchmarks werden auf dem Abstraktionsniveau von ausführbaren

Programmen (in einer Abfrage- oder Programmiersprache) angegeben, weil die zu bewertenden DBMSe gleiche oder ähnliche Programmierschnittstellen (z.B. SQL-Schnittstellen) zur Verfügung stellen. Da sich noch kein Standard für Schnittstellen von OMSen gebildet hat, der für die Entwicklung eines Benchmarks verwendet werden kann, müssen Benchmarks für OMSe abstrakter spezifiziert werden, so daß sie für eine Vielzahl von OMSen verwendbar sind.

Zur Reduzierung der Komplexität eines OMS-Benchmarks empfiehlt sich ein zweistufiges Vorgehen, wobei zunächst elementare OMS-Funktionen implementiert und gemessen werden, um dann komplexere Benchmark-Operationen unter Ausnutzung der gewonnenen Erfahrungen zu implementieren.

- Die praktische Brauchbarkeit der untersuchten OMS (-Prototypen) ist problematisch.

  Die Performance der Funktionen der Programmierschnittstelle (auf einer SUN 3/60) ist bei den meisten OMSen nur für einfache Applikationen ausreichend. Deshalb ist der OMS-Anwender gezwungen, sein Werkzeugdatenmodell auf einer groben Granularitätstufe zu modellieren. Hierdurch wird jedoch die Integration von verschiedenen Werkzeugen auf der Basis des OMS erschwert.

  Im Bereich der Administrationsfunktionen weisen alle untersuchten OMSe so gravierende Mängel auf, daß sie meist (d.h. bei üblichen Anforderungsprofilen) für ernsthafte Projekte unbrauchbar sind. Die Beseitigung dieser Mängel scheint oft konzeptionelle Änderungen der Datenmodelle und der Implementierung von OMS-Funktionen zu erfordern.

Diese Erkenntnisse werden in den folgenden Abschnitten genauer erläutert.

2 Ziele der Evaluierungen

Die Untersuchungen, von denen hier berichtet wird, wurden im Kontext des Projekts ESF (EUREKA Software Factory) durchgeführt. In dem Projekt ESF sollen existierende OMSe zur Konstruktion von SEU oder SEU-Komponenten verwendet werden. Unsere Evaluierungen verfolgten daher das ganz allgemeine Ziel, OMS-Anwender bei der Auswahl eines OMS zu unterstützen.

Bei der Bewertung von DBMSen (insb. von deren Performance) werden oft andere Ziele verfolgt wie: Vergleich von Implementierungen derselben Schnittstelle, Optimierung und Tuning eines DBMS oder Leistungsvorhersage für konkrete Lasten. Diese Ziele haben wir nicht verfolgt.

Eine Bewertung eines OMS basiert auf mehreren interessierenden Eigenschaften des OMS, für die einzelne Bewertungsergebnisse bestimmt werden können, sowie auf einem Verfahren, das die Einzelergebnisse schließlich zu einem Gesamtergebnis zusammenfaßt. Beispiele für Einzelergebnisse sind Meßwerte oder Noten auf Basis von vorgegebenen Notenskalen bzw. Bewertungskriterien.

Die Eigenschaften, nach denen potentielle SEU-Entwickler ein OMS bewerten, können zunächst grob eingeteilt werden in:

- technische Eigenschaften des OMS (Datenmodell, Architektur, Performance etc.) und
- nicht-technische Eigenschaften (Preis, Vertrauen in den Hersteller, Schulungsangebot, Verfügbarkeit, etc.)

Bei der Auswahl eines OMS sind nach unseren Erfahrungen von den technischen Eigenschaften die architektonischen Eigenschaften und der weite Bereich der Administrationsfunktionen oft wichtiger als das Datenmodell. Die technischen Eigenschaften sind bei der Auswahl wiederum meistens weniger relevant als nicht-technische Eigenschaften (dies mag aus akademischer Sicht bedauert werden, gilt aber paradoxerweise auch für akademische Objektmanagementsystem-Anwender).

Die Bewertung von nicht-technischen Eigenschaften hängt stark von den individuellen Anforderungen der Anwender ab, so daß wir sie nicht berücksichtigt haben.

3 Untersuchte Eigenschaften und Bewertungskriterien

Zunächst müssen mehrere Hauptkomponenten eines Objektmanagementsystems unterschieden werden:

- die Applikationsschnittstelle: ihre Realisierung (Bibliotheksfunktionen, Präprozessor o.ä.) ist beliebig. Die verfügbaren Funktionen können Datenmanipulation, Datendefinition und OMS-Administration betreffen.
- Standard-Browser bzw. Abfragesprachen für interaktive Benutzung
- Administrationswerkzeuge
- Programmierwerkzeuge bei OMSen, die die Definition von Datentypen mit typspezifischen Operationen erlauben

Untersucht haben wir vor allem folgende für die Brauchbarkeit eines OMS in einer SEU relevanten Eigenschaften:

1. Architektur, insb. Funktionsschichten, interne Schnittstellen, Verteilungskonzept, Erweiterbarkeit
2. Funktionalität der Applikationsschnittstelle, insb. das in ihr realisierte Datenmodell, Transaktionen, Zugriffsschutz, externe Sichten etc.
3. Performance der wichtigsten Datenmanipulationsoperationen
4. Funktionalität der Standard-Browser und der Administrationswerkzeuge

Für die Bereiche (1) bis (3) gibt es keine einheitlichen Bewertungskriterien. Dies ergibt sich daraus, daß ein OMS die Rolle der (bzw. einer) Datenhaltungskomponente innerhalb der Architektur einer SEU hat. Die Anforderungen, die ein OMS in dieser Rolle zu erfüllen hat, hängen stark von der SEU ab, in der das OMS eingesetzt werden soll. Dies erklärt zugleich, warum die Anforderungsdefinitionen für OMSe (es existiert rund ein Dutzend derartiger Dokumente, z.B. [GPI88]) und die vorhandenen OMSe derart unterschiedlich sind [ScH90]. Die wichtigsten relevanten Merkmale einer SEU bzgl. eines OMS sind:

- Relevante "äußere" (Leistungs-) Merkmale einer SEU:
  - Größenklasse der Projekte, z.B. anhand der Klassifikation in [PeK88] (Die Größenklasse hat großen Einfluß auf die folgenden Merkmale, ferner auf das Volumen der zu verwaltenden Daten.)
  - Anzahl der abzudeckenden Entwicklungsphasen bzw. Tätigkeiten, Anzahl der unterschiedlichen Methoden für einzelne Phasen
  - angestrebter Grad der Datenintegration aus Benutzersicht

  - Verteilung (keine, lokal oder weit)

- Relevante Architektur-Merkmale einer SEU:
  - angestrebte Form der Datenunabhängigkeit der Werkzeuge
  - angestrebte Form der Datenintegration (Integration über den Werkzeugentwurf, z.B. in SEUen, die i.w. aus Spezifikationen generiert werden, oder Integration über das OMS)
  - Granularität der Daten an der OMS-Schnittstelle
  - Offenheit, Erweiterbarkeit der SEU

Die einzelnen Merkmale können in einer Beziehung zueinander stehen. So kommen je nach den äußeren Merkmalen nur bestimmte Architekturen für SEUen in Frage.

Allgemeingültige Maßstäbe für die o.a. Eigenschaften (1) bis (3) gibt es also nicht; eine Bewertung ist höchstens auf Basis von Anforderungen, die durch SEU-Entwickler vorgelegt werden, möglich. Für die Eigenschaften der OMSe in diesen 3 Bereichen wurden daher keine absoluten Werturteile vergeben. Stattdessen wurde informell beschrieben, für welche Arten von SEU die einzelnen OMS geeignet erscheinen. Bei dem Bereich (4) hingegen können relativ allgemeingültige Kriterien aufgestellt werden. Hierzu wurden die Administrationsfunktionen, die in konventionellen DBMS und Dateisystemen vorhanden sind, anhand von Rollen analysiert und auf OMSe übertragen [DeK89, Ro89].

4 Benchmarks für Objektmanagementsysteme

"Abstrakte" Benchmarks. Benchmarks für Dateisysteme oder konventionelle DBMSe werden möglichst als Quellprogramm (z.B. in C oder SQL) angegeben. Dieses Vorgehen ist für OMS nicht anwendbar, weil deren Datenmodelle, Schnittstellen und Gastsprachen zu unterschiedlich sind. Benchmarks für Objektmanagementsysteme müssen stattdessen in dem Sinne abstrakt sein, daß sie eine Datenhaltungsaufgabe auf einem so hohen Abstraktionsniveau beschreiben, daß der Benchmark auf verschiedenartigen Objektmanagementsystemen implementierbar ist.

"Einfache" versus "komplexe" Benchmarks. Eine SEU wird typischerweise auf großen komplexen Datenbeständen arbeiten und über viele verschiedene OMS-Funktionen darauf zugreifen. Idealerweise sollte ein Benchmark für OMSe dieses Verhalten simulieren, also viele verschiedene komplexe (Anwendungs-) Operationen auf komplexen Datenstrukturen durchführen. Ein Beispiel für einen derartigen "komplexen" Benchmark ist der HyperModel Benchmark [BeAM88, BeAM90].

Komplexe Benchmarks sind aber insofern problematisch, als die Implementierung eines komplexen Benchmarks, insbesondere die Wahl der Datenstrukturen, für verschiedene Entwickler oder verschiedenene OMSe nicht immer reproduzierbar ist. Meistens existieren mehrere Alternativen für die Implementierung eines Benchmarks, die völlig verschiedene Laufzeiteigenschaften aufweisen. Deshalb kann nicht eine beliebige Implementierung gewählt werden, sondern die Implementierung muß optimiert werden, indem besonders geeignete Funk-

tionalitäten des OMS ausgenutzt und die Performance der verschiedenen Funktionen berücksichtigt wird. Letzteres führt zu dem *Paradoxon, daß man Informationen über die Performance eines OMS benötigt, um die Benchmarks zu implementieren, mit denen man derartige Informationen eigentlich erst gewinnen möchte.*

Zur Lösung dieses Optimierungsproblems empfiehlt sich ein zweistufiges Vorgehen, wobei zunächst die Performance von elementaren OMS-Funktionen an sehr einfachen Datenstrukturen gemessen wird. Wir benutzten hierzu einen "einfachen Benchmark" [De&90]. Anschließend werden komplexere Benutzeroperationen implementiert und gemessen.

Der "einfache" Benchmark. Die Datenstruktur des einfachen Benchmarks besteht aus drei verschiedenen Objekttypen (DIR, SMALL, BIG). Für die zwei Objekttypen SMALL und BIG sind jeweils drei Attribute vom Typ STRING definiert. Durch diese Attribute können Werte (d.h. Zeichenketten) mit einer maximalen Länge von 10 Byte, 80 Byte bzw. 160 Byte gespeichert werden ("kleine" Attributwerte). Zusätzlich ist für den Objekttyp BIG ein weiteres Attribut von Typ STRING definiert. Durch dieses zusätzliche Attribut können Werte (d.h. Zeichenketten) mit einer maximalen Länge von 10 KByte bzw. 128 KByte ("große" Attributwerte) gespeichert werden. Für den Objekttyp DIR sind keine Attribute definiert.

Für die Paare von Objekttypen DIR/BIG, DIR/SMALL und SMALL/BIG ist jeweils ein Beziehungstyp definiert worden. Für diese Beziehungstypen sind ebenfalls jeweils drei Attribute vom Typ STRING zur Speicherung von "kleinen" Attributwerten definiert.

Für die Ausführung des einfachen Benchmark wurde eine initiale Datenbank, die 3000 Objekten vom Typ SMALL und 400 Objekten vom Typ BIG enthält, erzeugt, so daß die Datenbank ca. 5 MB Nutzdaten enthält.

Die Operationen des einfachen Benchmarks umfassen sowohl Operationen auf Objekten mit "kleinen" Attributwerten als auch auf Objekten mit "großen" Attributwerten. Die Operationen des Benchmarks können wie folgt klassifiziert werden:

- Initialisieren, Öffnen und Schließen der DB
- Erzeugen und Löschen von Objekten ohne Initialisieren eines Attributwerts
- Schreiben und Lesen eines "kleinen" Attributwerts eines Objekts
- Schreiben und Lesen eines "großen" Attributwerts eines Objekts
- Schreiben und Lesen aller "kleinen" Attributwerte eines Objekts
- Erzeugen und Löschen von Beziehungen
- Schreiben und Lesen eines "kleinen" Attributwerts einer Beziehung

Erfahrungen. Die Erkenntnisse, die wir aus mehreren Implementierungen des einfachen Benchmarks und des HyperModel Benchmarks gewonnen haben [De&90, Ho&90], sind:

- Die Implementierung eines komplexen Benchmarks ist sehr aufwendig, z.B. für den HyperModel Benchmark sind ca. 2 - 4 Personenmonate pro OMS zu veranschlagen.
- Die Verwendbarkeit eines OMS in einer bestimmten SEU kann normalerweise schon anhand der Ergebnisse des einfachen Benchmarks entschieden werden. Der wesentlich höhere Implementierungsaufwand für einen komplexen Benchmark ist daher nur selten gerechtfertigt.

- Bei komplexen Benchmarks stellt die Vertrautheit der Implementierer mit dem OMS und die von ihnen erzielte Optimierung einen erheblichen Unsicherheitsfaktor dar.
- Die kontrollierte Durchführung der Messungen (gleiche Bedingungen für die Durchführung aller Benchmarks) und der damit verbundenen Konfigurierung der Rechner (Betriebssystem, Sekundär- und Hauptspeicher) ist häufig sehr aufwendig.
- Selbst beim einfachen Benchmark verursacht die kontrollierte Durchführung der Messungen einen erheblichen Aufwand; Werkzeuge zur Auswertung der Rohdaten sind unerläßlich.

Vergleich mit anderen Quellen. Bisher gibt es nur sehr wenig Literatur über Methoden zur Bewertung von OMSen und über konkrete Benchmarks für OMSe. Methoden zur Bewertung von (konventionellen) DBMSen oder Dateisystemen (z.B. [BiDT83, Tu87]) sind nicht ohne weiteres auf OMSe übertragbar. Diese Methoden beruhen wesentlich darauf, daß die Abfragesprachen und die Architektur dieser Datenverwaltungssysteme ähnlich oder sogar normiert sind; diese Voraussetzung ist bei OMSen nicht erfüllt.

Mit bisherigen Benchmarks für OMSe werden zwar auch elementare Funktionen gemessen werden [RuKC87, Ca88]. Diese Benchmarks sind allerdings stark durch relationale Systeme geprägt, d.h. der Aufbau der DB und die Art der Operationen sind völlig untypisch für viele Arten von Werkzeugen einer SEU: so fehlen Operationen zur Manipulation von Beziehungen (Beziehungen in konventionellen DBMSen werden durch Referenzen in Feldern eines Tupels modelliert) sowie Operationen zur Manipulation von langen Feldern.

Der schon erwähnte HyperModel Benchmark [BeAM88, BeAM90] wurde von einem Hypertext-System abgeleitet. Er ist als komplexer Benchmark einzuordnen und ist unseres Wissens nach der einzige bisher publizierte. Leider sind sowohl die Struktur der Daten wie auch viele der Operationen nicht typisch für viele Werkzeuge einer SEU.

5 Ergebnisse der Performance-Messungen

Für die Messungen wurden unterschiedliche Kombinationen der folgenden Bedingungen gewählt:

- mit und ohne Transaktionsschutz
- im "kalten" Zustand der DB (d.h. Lesen und Manipulation der Objekte jeweils direkt nach Öffnen der Datenbank) und "warmen" Zustand der DB (d.h. Lesen und Manipulation der Objekte jeweils sofort, nachdem das jeweilige Objekt gelesen wurde)
- mit verschieden großen DB-Inhalten.

Detaillierte Meßergebnisse können hier aus Platzgründen nicht wiedergegeben werden. Stattdessen werden die wichtigsten Ergebnisse exemplarisch beschrieben.

Die Operationen zur Manipulation von Objekten mit "kleinen" Attributen dauern meistens etwa 10 - 30 Millisekunden (Realzeit). Mit Transaktionsschutz erhöht sich diese Zeit z.T. beträchtlich. Nur das System GRAS, welches allerdings ein extrem einfaches Datenmodell hat, war deutlich schneller, allerdings auch nur bei kleinen Datenbanken, die komplett im Hauptspeicher gehalten werden können.

Operationen zur Manipulation von Objekten mit "großen" Attributen dauern im Vergleich zu äquivalenten Operationen auf Dateien

[EmK89] der gleichen Größe zwischen 2 und 20 Mal länger. Nur bei dem System PCTE/OMS zeigte sich keine signifikante Verlängerung.

Die Performance der untersuchten OMSe differiert bei manchen Operationen um mehrere Größenordnungen. Es ergibt sich allerdings kein einheitliches Bild zugunsten eines OMS; stattdessen sind unterschiedliche Ausrichtungen auf Architekturen und Optimierungen für Lastprofile erkennbar.

Die untersuchten OMSe sind für unterschiedliche Arten von Applikationen einsetzbar. OMSe mit "schwergewichtigen" Objekten (d.h. Objekte mit zusätzlichen Verwaltungsinformationen über sich selbst, wie z.B. Zugriffsrechte, letzte Zugriffszeiten, etc.) eignen sich z.B. eher für Konfigurationsmanagement-Werkzeuge in großen Projekten [PeK88], OMSe mit "leichtgewichtigen" Objekten (Objekte ohne Zusatzinformationen) eher für Syntaxeditoren in Programmierumgebungen.

Der Anwender eines OMS wird bei den meisten untersuchten OMSen gezwungen, sein Werkzeugdatenmodell auf einer groben Granularitätsstufe zu modellieren. Hierdurch wird die Integration verschiedener Werkzeuge auf der Basis eines OMS erschwert, da die Struktur der Daten in nichtstrukturierten Objekten verborgen bleibt und ein Abgleich der Objekte bzw. deren Strukturen nicht oder schwer möglich ist.

Keines der hier untersuchten OMSe ist auch nur annähernd schnell genug, um Objekte für Werkzeuge mit erheblichen Performance-Anforderungen (z.B. Petri-Netz-Simulatoren) zu verwalten. Für derartige Applikationen ist z.B. pro Sekunde die Ausführung von 10.000 oder mehr OMS-Operationen erforderlich (ähnliche Beobachtungen für CAD-Applikationen finden sich in [Ca88]). Es ist fraglich, ob derartige Leistungssteigerungen mit herkömmlichen Architekturen und Implementierungsmethoden erzielbar sind.

6 Ergebnisse der Bewertung der Administrationsfunktionen

Wie schon oben erwähnt treten hier generell gravierende Mängel auf. Die Beseitung dieser Mängel scheint oft Änderungen der Datenmodelle der OMSe zu erfordern. Dieser Abschnitt skizziert einige der Schwachpunkte und zeigt notwendige Weiterentwicklungen der OMSe auf.

Werkzeuge und Dokumentation. Die meisten OMSe verfügen über kein Administrationskonzept. Administrationshandbücher oder die Administration unterstützende Werkzeuge mit guten Benutzerschnittstellen sind die Ausnahme. Lediglich die Systeme DAMOKLES und PCTE/OMS unterstützen die Administration in ausreichendem Umfang.

Die Administrationswerkzeuge müssen in die SEU integrierbar sein. Eine der wichtigsten Regeln zur Konstruktion offener und erweiterbarer SEU lautet, die Funktionalität der Werkzeuge über eine Programmierschnittstelle verfügbar zu machen. Gegen diese Regel wird teilweise gröblichst verstoßen, indem sehr wichtige Funktionen nur über ein Benutzerinterface eines Programms verfügbar sind.

Recovery und Archivierung. In einer SEU hat man vor allem mit drei Arten von Datenverlusten zu rechnen:

1. Medienfehler
2. Inkonsistente DB infolge Systemabsturz
3. vom Benutzer versehentlich gelöschte oder zerstörte Objekte

Die meisten der untersuchten OMSe bieten nur rudimentäre Hilfen für die beiden ersten und keine Unterstützung für die letzte Fehlerart. Typischerweise kann nur die gesamte DB oder ein ganzes Segment auf eine Datei kopiert und somit auf Archivierungsmedien übertragbar gemacht werden. Dies ist viel zu grob und führt zu sehr großen Mengen von Backup-Daten. Lediglich eines der von uns untersuchten OMSe ermöglicht einen inkrementellen Dump, wie er in Dateisystemen üblich ist.

Erforderlich sind zusätzlich Funktionen, die es gestatten, aus Benutzersicht "sinnvolle" Einheiten des Backup bzw. der Archivierung zu definieren und diese auf entsprechende Medien und zurück zu übertragen. Diese Funktionen sollten mit dem Versionskonzept des OMS integriert sein. Besondere Probleme treten bei der späteren Wiedereinlagerung einer solchen Einheit in die Datenbank auf, weil sich die Umgebung, in der die Einheit früher existierte und mit der sie Beziehungen hatte, inzwischen verändert haben kann.

In der Praxis ist es unbedingt erforderlich, daß alle Recovery-Daten entweder automatisch oder durch einen (zentralen) Operator erzeugt werden können und daß Benutzer der SEU hiermit nicht belästigt werden (dies ist bei konventionellen DBMS und Dateisystemen völlig selbstverständlich). Die untersuchten OMSe erlauben dies nicht, weil z.B. Schema-Informationen von außen nicht zugänglich sind (keine Meta-DB) oder weil sie eine eigene Benutzer- und Rechteverwaltung haben, die mit derjenigen des unterliegenden Betriebssystems nicht konsistent ist.

Ein spezielles Problem bei OMSen entsteht durch die Typisierung von Objekten und durch die vergleichsweise dynamische Entwicklung der Typen (bzw. Schemata). Es muß möglich sein, zusammen mit Daten die zugehörigen Schemata zu restaurieren. Diese Anforderung wird von keinem der untersuchten OMSe erfüllt.

Standard-Browser. Leistungsfähige Browser sind völlig unerläßlich und fehlen bei fast allen OMSen. Es sollte wenigstens je ein Browser für alphanumerische und graphische Terminals angeboten werden. Da man sich bei navigierenden Datenmodellen leicht in der Datenbank "verläuft", sollte man durch einen Browser geeignet unterstützt werden.

Datenaustausch. Man ging vor einigen Jahren von der Vorstellung aus, daß ein OMS das einzige Datenverwaltungssystem in einer SEU sein würde. Dies wird wahrscheinlich eine Illusion bleiben. Hieraus ergibt sich die Notwendigkeit, OMSe mit Dateisystemen und konventionellen DBMSen zu integrieren. Diese Aufgabe wird von den heutigen OMSen noch nicht unterstützt.

7 Ausblick

In der nächsten Projektphase sollen weitere OMSe bewertet werden. Ferner wird der einfache Benchmark überarbeitet. Die bisherigen Erfahrungen zeigen, daß einige darin definierte Operationen entfernt werden können, ohne daß dies die Aussagekraft der Ergebnisse der Benchmarks wesentlich einschränkt.

Zusätzlich wird an einem komplexen Benchmark im Hinblick auf die Modellierung von komplexen Objekten und den Zugriff auf komplexe Objekte oder Teile eines komplexen Objekts gearbeitet, der die oben erwähnten Nachteile des HyperModel Benchmarks nicht aufweist.

Literatur

[BeAM88] Berre, A.J.; Anderson, T.L.; Mallison, M.: VBase and the HyperModel Benchmark; Oregon Graduate Center, Dep. Computer Science and Engineering, Technical Report No. CS/E-88-031; 1988

[BeAM90] Berre, A.J.; Anderson, T.L.; Mallison, M.: The HyperModel Benchmark; p. 317-331 in: Bancilhon, F. et. al. (ed.): Database Technology EDTB 90; LNiCS 416, Springer; 1990

[BiDT83] Bitton, D.; DeWitt, D.J.; Turbyfill, C.: Benchmarking database systems - A systematic approach; p.8-19 in: Proc. VLDB 83; 1983/11

[Ca88] Cattell, R.G.G.: Object-oriented DBMS performance measurement; p.364-367 in: Dittrich, K.R. (ed.): Advances in object-oriented database systems; LNiCS 334, Springer; 1988

[De&90] Dewal, S.; et.al.: Evaluation of object management systems; SWT-Memo 44, Universität Dortmund, ISSN 0933-7725; 1990

[DeK89] Dewal, S.; Kelter, U.: Role-based requirement definition for software factories using reusable requirement packages; p. 351- 370 in: Bennet, K.H. (ed.): Software Engineering Environments: Research and Pratice; Ellis Horwood Limited; 1989

[EmK89] Emmerich, W.; Kelter, U.: File system benchmarks on workstations; SWT-Memo 41, Universität Dortmund, ISSN 0933- 7725; 1989

[GPI88] German PCTE Initiative: Requirements for the enhancement of PCTE/OMS, Version 2.0; 1988/09

[Ho&90] Hormann, H.; Platz, D.; Roschewski, M; Schöpe, L.: The HyperModel Benchmark: description, execution and results; SWT-Memo 53, Universität Dortmund, ISSN 0933-7725; 1990

[PeK88] Perry, D.E.; Kaiser, G.E.: Models of software development environments; p.60-68 in: Proc. IC Software Engineering 88; 1988

[Ro89] Roschewski, M.: Anforderungsanalyse und Bewertung von Administrationsfunktionen in Objektmanagementsystemen; Diplomarbeit, Universität Dortmund; 1989

[RuKC87] Rubenstein, W.B.; Kubicar, M.S.; Cattell, R.G.G.: Benchmarking simple database operations; p.387-394 in: Proc. SIGMOD 87; 1987

[ScH90] Schöpe, L.; Hormann, H.: Übersicht über Nicht-Standard-Datenbanksysteme; SWT-Memo 42, Universität Dortmund, ISSN 0933- 7725; 1990

[Tu87] Turbyfill, C.: Comparative benchmarking of relational database systems; Dissertation, Dep. Computer Science, Cornell Univ., TR 87-871; 1987/09